On the performance analysis of distributed caching systems using a customizable Markov chain model

Hamza Ben Ammar, Yassine Hadjadj-Aoul, Gerardo Rubino, Soraya Aït-Chellouche

On the performance analysis of distributed caching systems using a customizable Markov chain model

Hamza Ben-Ammar, Yassine Hadjadj-Aoul, Gerardo Rubino and Soraya Ait-Chellouche
Univ Rennes, Inria, CNRS, IRISA
Emails: {hamza.ben-ammar, yassine.hadjadj-aoul, gerardo.rubino, soraya.ait-chellouche}@irisa.fr

Abstract

In the last few years, Networks Operators (NO) have experienced an increased number of requests for video contents and rich media services, which are becoming increasingly popular. In view of the network scaling limitations, operators are developing their own caching systems to speed up the network performance. Indeed, disseminating caches in the infrastructure not only helps in absorbing the network's congestion, but in addition, brings content closer to users, which allows a reduced latency. Several studies have focused on improving the performance of such caching systems, especially in the context of Content-Centric Networking (CCN). In this paper, we propose a fairly generic model of caching systems that can be adapted very easily to represent different caching strategies, even the most advanced ones. Indeed, the proposed model of a single cache, named MACS, which stands for Markov chain-based Approximation of CCN Caching Systems, can be extended to represent an interconnection of caches under different schemes. In order to demonstrate the accuracy of our model, we proposed to derive the two most effective techniques in the literature, namely LCD and LRU-K, which may adapt to changing patterns of access. Simulation results using a discrete event simulator clearly show the accuracy of the proposed model under different network configurations.

Keywords: Content-Centric Networking, Multi-cache systems, Caching, Markov chain, Modeling.

1. Introduction

The rapid growth of media-enriched services, over the past few years, has significantly changed the way that people experience the Internet, making media traffic, and especially video traffic, one of the main drivers of the Internet Economy [1]. At the delivery infrastructure level, this manifests as a huge need of storage, processing resources and, more critically, bandwidth capacities.

Such data consumption context allowed CDNs to be at the center of the content distribution value chain. ISPs, for their part, are struggling to benefit from this traffic increase. On the contrary, this trend incurs large expenditures to meet the increasing demand and satisfy subscribers. Besides, given the adopted flat rate business models, the Average Revenues per Users (ARPU) are getting lower. ISPs are, thus, investigating the possibilities to extend their infrastructure with caching capabilities as a way to be part of this content distribution value chain. The advent of Content-Centric Networking (CCN) represents, thus, a real opportunity for ISPs [2]. In fact, CCN networks enable focusing on the content itself and not on its location, which allows to overcome the limitations of the current Internet network. Thus, end-users' requests that are routed toward the Content Providers' servers, can be satisfied by intermediate caching nodes, which allows reducing the network traffic and the servers' loads while shortening the latency for end-users.

When content caching is possible, a significant improvement can be achieved, as shown in several studies [3]. The analytical quantification of caching performance is, however, not sufficiently explored in the literature. In fact, several issues need to be addressed in order to understand the behavior of such a caching network. Indeed, in addition to the cache hit probability, other metrics deserve a deeper analysis, such as the average number of hops to reach a particular content, or the impact of cache size and traffic pattern on performance.

In a previous work [4], we proposed an analytical model based on Markov chains named MACS (Markov chain-based Approximation of CCN Caching Systems). This model allows us to estimate the cache hit probability under the popular Least Recently Used (LRU) replacement scheme for a system with multiple caching nodes, where the default caching strategy of CCN, called Leave-Copy-Everywhere (LCE), is used. LCE consists in caching the requested contents in all the nodes along the downloading path. Then, we extended MACS in [5] and proposed a methodology that allows modeling the caching decision process in a more general way, so that it can be used to analyze different caching algorithms other than LCE.

In this paper, we investigate further our model and show how its versatility allows it to mimic the most efficient caching strategies like Leave Copy Down (LCD) [6] and 2Q [7]. We aim with our model to gain more insights

on the efficiency of CCN caching strategies by analyzing different network performance metrics like cache hit rate, content provider load and distance reduction ratio.

We consider in this work a network of caches in which the requests arrive independently and follow a popularity law (i.e. Zipf), which is generally the case in real systems [8]. All the requests are forwarded through the shortest path to the nearest source. We start by proposing a discrete time Markov chain to model a single cache node. This model is, then, generalized to a system of caches. Compared to most of the existing contributions in the literature, the proposed solution can easily deal with different network topologies and allows having an accurate approximation of such type of caching systems.

The reminder of this paper is organized as follows. In section 2, we present the related work, which focuses on contributions providing analytical models of caching systems. Section 3 provides the detailed description of the proposed model of a single cache. The case of multiple caches system is described in section 4. Then, the evaluation of our model is introduced in section 5. Finally, section 6 summarizes the achievements of this paper and introduces our future work.

2. Related Work

Many studies have been conducted these last years to deal with the performance analysis of a single cache and a network of caches [9] [10].

The study, in [11], was probably the first attempt to evaluate and model caching systems. The author proposed a model for predicting the buffer hit probability under the LRU and FIFO replacement policies. Unfortunately, the computational complexity of this model grows exponentially with the cache size C and the number of data items R. In [12], Dan and Towsley proposed an algorithm with a complexity of $O(CR)$ to predict an approximate cache hit probability under the LRU replacement policy. The proposed solution is, however, limited to the case of a single cache.

In [13], the authors extended the algorithm of [12] to the case of a network of caches. Che et al. developed, in [14], an analytical modeling technique, which was further investigated in [15]. The solution allows identifying a characteristic time approximation for each item in the cache, which was used to estimate the cache hit rate per content. In [16], the authors extended the work of [14] and proposed Time-To-Live (TTL) based caching model that assigns a timer to each content stored in the cache and redraws it every time the content is requested. Psaras et al. proposed, in [17], a Markov chain-based caching model to estimate the proportion of time a given piece of content is cached in the case of a single router. They also extended their model to cover the case of multiple caching nodes. Nevertheless, the aforementioned proposals are applicable only when the LCE caching strategy is used.

More recently, there have been contributions to improve CCN networks by using alternative caching strategies to the LCE. In [18], the authors studied the probabilistic caching strategy where the caching decision probability is a prefixed parameter to explore its behavior with different cache replacement policies (including LRU). However, their work was done only through computer simulation and limited network settings were considered. Tarnoi et al. developed, in [19], an analytical model based on Markov chains to compute the cache hit rate under the probabilistic scheme and with different cache replacement policies (including LRU). They evaluated their proposal through simulation, but the tests were done only in the case of a single cache and a two-nodes' network with a small-sized catalog of contents. Note that probabilistic caches give much less performance than some more advanced techniques such as LRU-K, which will be presented later.

In [20], the authors extended the work of [14] and proposed a unified framework to analyze the performance of caches (both isolated and interconnected). Their model covers various insertion and eviction policies (including LRU and LCD). They evaluated the accuracy of their proposal through simulation. However, in the case of interconnected caches, the tests were conducted only in the case of a 6-nodes chain with fixed network settings. The authors in [21] developed algorithms to approximate the hit probability of the cache replacement policy LRU-K [22] [7] and variants of it. Nevertheless, their proposal is limited only to the case of a single cache.

In this paper, we develop an analytical model to address the performance evaluation of multi-cache systems under the LRU replacement policy and different caching schemes. The proposed model is also validated using computer simulations under various network configurations. The following section presents our proposal with all the needed information and the adopted assumptions.

3. A general Markov chain model of a single cache

In the present work, the LRU algorithm is used to manage the node's content store. Other memory management algorithms have been studied in the case of cache modeling like First In First Out (FIFO) and Random Replacement (RR). However, LRU is known to perform much better than FIFO [23] and the expected long-run performances of both RR and FIFO replacement policies were shown to be the same [24]. The Least Frequently Used (LFU) algorithm, which requires a more complex replacement process, is also interesting. However, it presents several shortcomings related to the insertion of new popular contents or the purge of older ones [25]. Moreover, the complexity of the LFU functioning makes it harder to be modeled and analyzed.

3.1. System description

In CCNs, the content's name is the only identifier of data. To request and retrieve data, two types of packets

Table 1: Summary of the notations.

Term	Desciption
M	Number of nodes in the network
R	Number of items in the catalog
N	Cache size
S	State space of an LRU cache model
c_r	Content with a popularity/rank r
p_r	Probability to request an item c_r
p'_r	Probability that a node receives an item c_r in a multi-cache network
α	Zipf law distribution parameter
$\beta(r)$	Cache decision probability
$\gamma'_s(r)$	Probability of a content c_r staying at the same position s of the cache
$\pi(r)$	Steady state distribution of the Markov chain
$P_{hit}(r)$	Hit probability of content c_r
$P_{miss}(r)$	Miss probability of content c_r
$MS(r,u)$	Outgoing miss stream ratio from a node u for a content c_r
$\eta_r(v)$	Incoming miss stream ratio at a node v for a content c_r
$Dist(i)$	Distance from where the clients requests were generated to the node v_i

are commonly used [2]: Interest Packet and Data Packet. Clients can ask for specific data objects by sending Interest packets, which are forwarded towards the data sources using the Forwarding Information Base (FIB). A record of the forwarded Interests is kept in the Pending Interest Table (PIT) in order to keep track of the Interests waiting for a data packet. When a node receives multiple requests for the same content, the Interest packets will be aggregated in one entry in the PIT and only the first one is routed. Once the requested content is found, it is automatically routed back to the clients on the reverse path. All the nodes along this path can store a copy of data to answer future demands.

Let $G = (V, E)$ be the graph representing a general network of caches, where $V = \{v_1, ..., v_M\}$ depicts the nodes of the network and $E \subset V \times V$ is the set of links connecting the nodes. Each node in the network is equipped with a caching module used to store contents locally. Let $C = \{c_1, ..., c_R\}$ be the set of the catalog's contents available for the users. We assume that all the accessible contents in the system have an identical size and are divided into small packets or chunks, which are in turn of the same size. The cache capacity is then expressed in terms of the number of contents or chunks that can be stored. All the available contents are stored permanently at one or more servers attached to some nodes within the network. In the rest of the paper and for the sake of readability, we will use the term node/cache interchangeably as well as the terms rank/popularity and content/item/object.

Clients, which are attached to the network nodes, send requests into the network looking for contents. The pattern of these requests is characterized by the Independent Reference Model (IRM) [12]. Suiting the IRM model, users generate an independent and identically distributed sequence of requests from the catalog C of R objects. Specifically, the probability p_r to request an item c_r from the set of available contents in the network is constant and follows a popularity law, where the contents are ranked decreasingly according to their popularity from one to R. Since, in our work, we address video services, the contents feature a skewed popularity distribution. As already argued in many previous studies, the latter fits the Zipf law [26]: the probability to request the content of rank r is: $p_r = r^{-\alpha} / \sum_{i=1}^{R} i^{-\alpha}$, where α, the skew of the distribution, depends on the type of the accessible objects [15]. For our approach, we only need to assume that $0 < p_r < 1$, with $R \geq 2$.

3.2. A comprehensive analysis of LRU caches

Let's consider a single node in a Content-Centric Network operating under the LRU replacement policy and having clients attached to it. Whenever a user requests a content of rank r in the catalog, it will generate either a cache miss if the content is not present in the cache or a hit otherwise. In the latter case, the object will be sent back to the user. In the case of a cache miss, the client's interest is forwarded to the next nodes in the direction of the nearest content server storing a permanent copy (i.e. origin server). Once the object located, it is sent on the reverse path and depending on the caching strategy used in the network, the content will be cached or not at each node that passes by it. In CCN, a node uses a caching scheme in order to decide whether an incoming item should be saved or not. In the general case, this decision can be associated with the probability that we denote $\beta(r)$, with which a received content c_r will be stored in a cache. The value of $\beta(r)$ will then depend on the caching strategy adopted by the node. When using the LCE policy, for example, all arriving objects are cached, that is, $\beta(r) = 1$ for any content c_r. We will see later in more detail the values taken by $\beta(r)$ when a specific caching strategy is used.

Consider a cache sized to contain $N \leq R$ items (the usual case is, of course, $N \ll R$). Whenever a local cache hit or a caching decision occurs for a content c_r (the latter with probability $\beta(r)$), it will then be placed at the top position in the cache. Consequently, for any given content $c_{r'}$ occupying position i in the cache, with $r' \neq r$, three actions are possible when a request for c_r is received:

- $c_{r'}$ will be moved down by one position if the requested content c_r is either outside the cache and it has been decided to cache it (with probability $\beta(r)$) or if it occupies block j with $j > i$;
- $c_{r'}$ will remain at the same position if c_r occupies a block j with $j < i$, or if it is outside the cache and

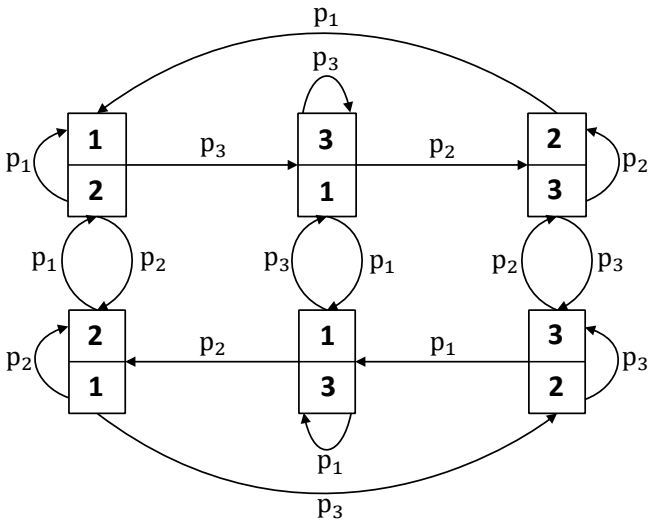

Figure 1: A Markov chain model of an LRU cache where the LCE strategy is adopted in the case of $R = 3$ and $N = 2$.

it has been decided not to cache it (with probability $1 - \beta(r)$);

- $c_{r'}$ will be evicted from the cache if it occupies the N^{th} position (last block) and if it has been decided to cache c_r (with probability $\beta(r)$), which is outside the cache.

When analyzing the performances of a caching system, we usually focus on hits or misses, which means that we consider the system only when requests for content arrive. This means that in case of a miss, we don't look at the time needed to get a copy of that content in the cache; we only consider which contents are in which positions at arrival times.

Let us call a *configuration* of the cache any vector $\vec{x} = (x_1, \ldots, x_N)$ where x_i is the content present at block or position i. We only consider the case where all positions are occupied because the cases where the cache is partially filled concerns only the initial transient phase. Let us denote by S the set of all possible configurations; we have $|S| = R!/(R-N)!$. For realistic values of N and R, $|S|$ is huge. If we consider the evolution of the configuration of the cache "just after" the arrival of a request, we define a discrete time homogeneous Markov chain $X = (X_n)$, where X_n is the configuration just after the arrival of the n^{th} request at the node. Chain X is clearly irreducible: from any configuration $(x_1, \ldots, x_N)$, we can move to any other configuration $(y_1, \ldots, y_N)$ after the arrival of a request for content y_N, then for content y_{N-1}, etc. (this event has probability $p_{y_1} \cdots p_{y_N} > 0$). Of course, we consider the case of always deciding to store the arriving content. It is also aperiodic because if $X_n = (x_1, \ldots, x_N)$, then, with probability p_{x_1} next state is still $(x_1, \ldots, x_N)$. So, X is ergodic.

Knowing the steady state distribution of X, $\left(\pi_{\vec{x}}\right)_{\vec{x} \in S}$, allows (in theory) to evaluate important performance metrics of a caching system such as the hit probability per content. Assuming the system is in equilibrium, we have that for any $c_r \in C$,

$$\Pr(\text{hit for a request of } c_r) = \sum_{\vec{x} \text{ s.t. } \exists i \in \{1,\ldots,N\} | x_i = c_r} \pi_{\vec{x}}. \quad (1)$$

For illustration purposes, let us consider the case of $R = 3$ and $N = 2$, when the content is always cached when received. The state space is $S = \{(1,2),(1,3),(2,1),(2,3),(3,1),(3,2)\}$ (see Figure 1). The stationary distribution $\pi_{\vec{x}}$ satisfies the equations

$$\begin{cases} \pi_{(i,j)} = p_i(\pi_{(i,j)} + \pi_{(j,i)} + \pi_{(j,k)}), \quad k \notin \{i,j\}, \\ \sum_{(i,j) \in S} \pi_{(i,j)} = 1. \end{cases} \quad (2)$$

Solving the equations, we have that for a generic configuration $(i,j) \in S$,

$$\pi_{(i,j)} = p_i p_j (1 - p_i)^{-1}.$$

If we want to compute the hit probability of content c_1, we obtain

$$\begin{aligned} \Pr(\text{hit for a request of } c_1) &= \pi_{(1,2)} + \pi_{(1,3)} + \pi_{(2,1)} + \pi_{(3,1)} \\ &= \frac{p_1(1 - p_2 p_3)}{(1-p_2)(1-p_3)}. \end{aligned}$$

We can see through this example how the steady state distribution of the Markov chain representing the dynamics of an LRU cache can be used to evaluate important performance metrics of the caching system. However, the problem with the exact analysis is the huge size of the state space S and the complexity of the model. In the next section, we describe an efficient way of obtaining an approximation for such a metric that is shown to be very accurate by comparing its output to the result of simulations.

3.3. A generic model of a single LRU cache

In [4], we modeled the caching strategy where every data packet is cached at every node that passes by it, which matches the case where the caching decision probability is always equal to one. The model was then extended in [5] to cover the general case of caching decision. In this paper, we investigate further our proposal MACS and show how it can be applied to model different caching strategies. It has to be noted that the flexibility of our model concerns only the caching decision process and not the cache replacement policy.

Let us consider a Markov chain $X(r) = \left(X_h(r)\right)_{h \geq 0}$ with $N+1$ states, as depicted in Figure 2. This chain represents the evolution in time of the position occupied by content c_r in the cache, where state $N+1$ means that content c_r is absent, state 1 means that the object is at the top of the

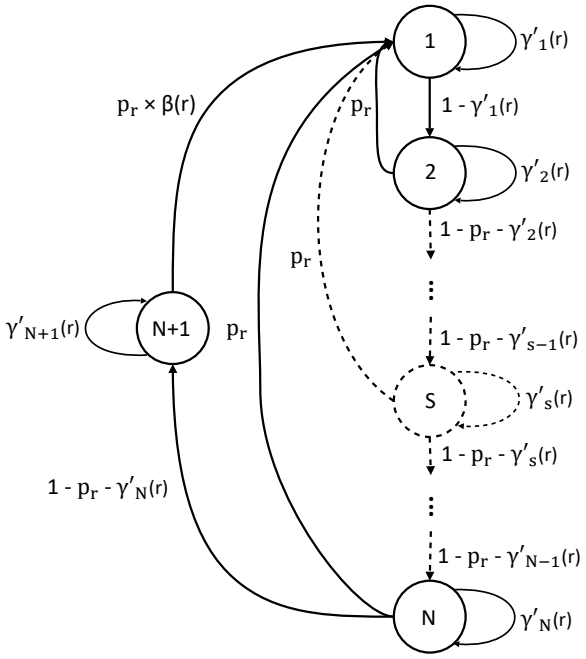

Figure 2: A generic Markov chain model for a content c_r in an LRU cache.

cache and state N means that it is at the bottom. The probability of a content staying at the same position upon the reception of an item is represented by $\gamma'_s(r)$.

Assume we know $\gamma'_i(r)$, $i = 1, 2, \ldots, N, N+1$, satisfying $0 < \gamma'_i(r) < 1$. This means that $X(r)$ is irreducible and aperiodic. Let us denote by $\pi(r) = (\pi_1(r), \pi_2(r), \ldots, \pi_{N+1}(r))$ its equilibrium distribution. Assume now that $\pi(r)$ is exactly the marginal distribution corresponding to content c_r and that the chains $X(1), \ldots, X(R)$ are independent of each other. The probability that a content of rank r remains in state s of the chain, with $s \in \{1, 2, \ldots, N+1\}$, upon the arrival of a request, is equal to

$$\begin{cases} \gamma'_s(r) = \gamma_s(r) + \sum_{i=1, i\neq r}^{R} (1-\beta(i))\ p_i\ P_{miss}(i), 1 \leq s \leq N, \\ \gamma'_{N+1}(r) = 1 - p_r\ \beta(r). \end{cases} \tag{3}$$

The value of $P_{\text{miss}}(i)$ used in (3) represents the cache miss probability of content c_i (its calculation will be detailed later) and the value of $\gamma_s(r)$ is a special case of $\gamma'_s(r)$ where $\beta(i) = 1$, and it is given by

$$\begin{cases} \gamma_1(r) = p_r, \\ \gamma_s(r) = \sum_{i=1, i\neq r}^{R} p_i \sum_{j=1}^{s-1} \pi_j(i), \qquad 2 \leq s \leq N, \\ \gamma_{N+1}(r) = 1 - p_r. \end{cases} \tag{4}$$

The first term of the expression of $\gamma'_s(r)$ (i.e. $\gamma_s(r)$) represents the case where a content c_r remains at the same position i when the received item occupies a position j, with $j < i$. The second term of $\gamma'_s(r)$ (i.e. $\sum_{i=1, i\neq r}^{R} (1-\beta(i))\ p_i\ P_{miss}(i)$) corresponds to the event where the received content is not in the cache and it has been decided to discard it. Again, observe that (3) is an approximation, but as we will see, its form makes that it leads to a very accurate approximation algorithm. Let's denote by $T_{i,j}$ the transition probability from state i to j in our model. Then, $T_{i,j}$ is defined as follows:

$$\begin{cases} T_{i,i} = \gamma'_i(r), \qquad 1 \leq i \leq N+1, \\ T_{i,1} = p_r, \qquad 2 \leq i \leq N, \\ T_{N+1,1} = p_r\ \beta(r), \\ T_{1,2} = 1 - \gamma'_1(r), \\ T_{i,i+1} = 1 - p_r - \gamma'_i(r), \qquad 2 \leq i \leq N, \\ T_{i,j} = 0, \quad j \notin \{1, i, i+1\}. \end{cases} \tag{5}$$

The distribution $\pi(r)$ satisfies the Chapman-Kolmogorov equations:

$$\begin{cases} \pi_i(r) = \sum_{j=1}^{N+1} \pi_j(r) T_{j,i}, \qquad 1 \leq i \leq N+1, \\ \sum_{i=1}^{N+1} \pi_i(r) = 1. \end{cases} \tag{6}$$

If we develop the system of equations defined in (5) and (6), we obtain

$$\begin{cases} \pi_1(r) = \gamma'_1(r)\pi_1(r) + p_r \sum_{i=2}^{N} \pi_i(r) + p_r\beta(r)\pi_{N+1}(r), \\ \pi_2(r) = \gamma'_2(r)\pi_2(r) + (1 - \gamma'_1(r))\pi_1(r), \\ \pi_3(r) = \gamma'_3(r)\pi_3(r) + (1 - p_r - \gamma'_2(r))\pi_2(r), \\ \ldots\ldots = \ldots\ldots \\ \pi_N(r) = \gamma'_N(r)\pi_N(r) + (1 - p_r - \gamma'_{N-1}(r))\pi_{N-1}(r), \\ \pi_{N+1}(r) = \gamma'_{N+1}(r)\pi_{N+1}(r) + (1 - p_r - \gamma'_N(r))\pi_N(r), \\ \pi_1(r) + \pi_2(r) + \ldots + \pi_{N+1}(r) = 1. \end{cases} \tag{7}$$

By computing $\pi_{N+1}(r)$, the cache miss probability, we can obtain the cache hit rate $1 - \pi_{N+1}(r)$. We can see from (7) that each $\pi_i(r)$ depends on $\pi_{i-1}(r)$ $(2 \leq i \leq N+1)$, and once we get $\pi_1(r)$, we can calculate $\pi_{N+1}(r)$ and deduce the cache hit. When $\beta(r) = 1$, we have $\pi_1(r) = p_r$. The equations then derived in (7) can be easily solved using (4) to finally get

$$\begin{cases} \pi_1(r) = p_r, \\ \pi_2(r) = \dfrac{p_r(1-p_r)}{1-\gamma_2(r)}, \\ \pi_i(r) = \dfrac{p_r(1-p_r)\prod_{j=2}^{i-1}(1-p_r-\gamma_j(r))}{\prod_{j=2}^{i}(1-\gamma_j(r))}, 3 \leq i \leq N+1, \\ \pi_1(r) + \pi_2(r) + \ldots + \pi_{N+1}(r) = 1. \end{cases} \tag{8}$$

Algorithm 1 Content's hit rate in a single cache

function $Get\_Phit(r, \beta(r))$
  $\pi'_{N+1}(r) \leftarrow Get\_Pmiss(r, 1)$
  $//\pi'_{N+1}(r)$: Beginning of the fixed-point iteration ($\beta(r) = 1$)
  $\pi_{N+1}(r) \leftarrow Get\_Pmiss(r, \beta(r))$
  ϵ : Arbitrarily small positive quantity
  while $|\pi_{N+1}(r) - \pi'_{N+1}(r)| \geq \epsilon$ **do**
    $\pi'_{N+1}(r) \leftarrow \pi_{N+1}(r)$
    $\pi_{N+1}(r) \leftarrow Get\_Pmiss(r, \beta(r))$
  end while
  return $1 - \pi_{N+1}(r)$
end function

In the general case ($0 \leq \beta(r) \leq 1$), $\pi_1(r)$ cannot be computed directly, and it depends on all the other values of $\pi_i(r)$ (for $2 \leq i \leq N+1$). One way to resolve this problem is to modify the expression of $\pi_1(r)$ in order to reduce its dependency on the other variables. To do so, we first add and subtract in the expression of $\pi_1(r)$ the value $p_r\ \pi_{N+1}$. Then, using the expression of $\gamma'_1(r)$ and since $\sum_{i=1}^{N+1} \pi_i(r) = 1$, we get a new expression of $\pi_1(r)$:

$$\pi_1(r) = p_r + (\gamma'_1(r) - \gamma_1(r))\pi_1(r) + p_r(\beta(r) - 1)\pi_{N+1}(r).$$

The expression of $\pi_1(r)$ can be rewritten as

$$\pi_1(r) = \frac{p_r(1 + (\beta(r) - 1)\pi_{N+1}(r))}{1 - (\gamma'_1(r) - \gamma_1(r))}. \tag{9}$$

Now, $\pi_1(r)$ depends only on $\pi_{N+1}(r)$. The idea then, is to use a fixed-point iteration in order to generate successive approximations of the solution, which consists on finding $\pi_{N+1}(r)$, by starting from an initial guess. To do so, we start by considering an approximate value of $\pi_{N+1}(r)$ (that we denote by $\pi'_{N+1}(r)$) by using the one obtained when $\beta(r) = 1$, to compute $\pi_1(r)$. Once we have $\pi_1(r)$, we can calculate $\pi_i(r)$ ($2 \leq i \leq N+1$) to finally obtain a new value of $\pi_{N+1}(r)$.

$$\begin{cases} \pi_1(r) = \dfrac{p_r(1 + (\beta(r) - 1)\pi'_{N+1}(r))}{1 - (\gamma'_1(r) - \gamma_1(r))}, \\ \pi_2(r) = \gamma'_2(r)\pi_2(r) + (1 - \gamma'_1(r))\pi_1(r), \\ \pi_i(r) = \dfrac{(1 - p_r - \gamma'_{i-1}(r))\pi_{i-1}(r)}{1 - \gamma'_i(r)}, 3 \leq i \leq N+1, \\ \sum_{i=1}^{N+1} \pi_i(r) = 1. \end{cases} \tag{10}$$

The value of $\pi'_{N+1}(r)$ is also used to calculate $\gamma'_s(r)$ ($s \in 1, ..., N$) by replacing $P_{miss}(r)$ with $\pi'_{N+1}(r)$. These steps are repeated until we converge to the final solution by replacing in each step $\pi'_{N+1}(r)$ by the new computed value $\pi_{N+1}(r)$ (see Algorithm 1). As for the computational complexity of our model, let's recall first that the state space contains $N+1$ elements. The complexity of computing the cache hit of one content c_r is then $O(N)$ (we suppose here that the complexity of calculating each $\pi_i(r)$, $1 \leq i \leq N+1$, is a constant). To compute the total cache hit, we have to apply MACS to each content of the catalog. Since we have R available items in the catalog, then the complexity of computing the average hit of a single cache is $O(NR)$. Now, in case where we have a multi-cache system containing M nodes and if we want to compute the average hit ratio of all the caches, we obtain a complexity of $O(NRM)$.

3.4. Single cache model for 2Q

In [22], the authors proposed LRU-K, a page replacement algorithm for database disk buffering. The proposal is an enhancement of the classical LRU replacement policy in the sense that a requests' history is maintained for the elements of the cache. Indeed, LRU-K keeps track of the timing of the K last occurrences, which allows having a better idea of the contents' popularity. Thus, the element whose K^{th} most recent access is furthest in the past will be evicted when the cache is full. When K is equal to one (LRU-1), the approach is equivalent to the classical LRU. Note that, most of the LRU-K method gain is achieved when $K = 2$ (LRU-2) [22]. However, LRU-2 suffers from a high complexity, as each element access requires $\log(N)$ operations to manipulate a priority queue (where N is the cache size).

Johnson et al. proposed the 2Q scheme [7], which is similar and performs as well as LRU-2 algorithm, but having a constant time overhead. Instead of cleaning cold elements from the main buffer like LRU-2, 2Q admits only the hot ones to the cache. When a request is received by a cache using 2Q, the requested object's hash is first placed in a virtual cache (called $A1$), which is managed as a FIFO. If an item is requested during its $A1$ residency, it is then promoted to the main cache (called Am). The authors, then, proposed another version of 2Q in which the $A1$ queue is partitioned into $A1in$ and $A1out$. The $A1in$ queue along with Am form the physical cache and $A1out$ is a virtual cache, which will contain only items hashes. The most recent first accesses will be stored in $A1in$, which will be managed as a FIFO queue. When objects are evicted from $A1in$, they will be remembered in $A1out$. Upon arrival of a request and if it is present in the $A1out$ queue, then it is cached in Am. The item to be discarded in 2Q is chosen either from $A1in$ or Am. However, the sizes of the different queues ($A1in$, $A1out$ and Am) are sensitive to the requests patterns and should be tuned carefully.

In our work, we considered a caching scheme similar to the first version of 2Q where the virtual buffer and the main cache are both managed using LRU. If we apply the model MACS presented previously to the 2Q scheme, $\beta(r)$ will then be equal to the hit probability in the virtual cache of the received content c_r. Since the virtual cache is managed as a classic LRU (i.e. each received item is always cached), $\beta(r)$ of a node v is obtained using equations (8):

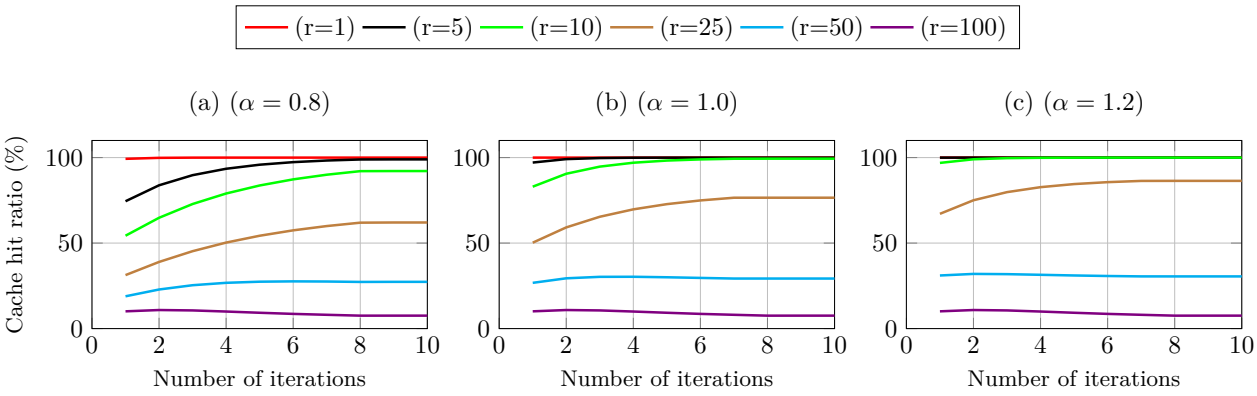

Figure 3: Cache hit ratio vs number of iterations in the 2Q fixed-point iteration of contents with various popularities under a single node (catalog = 20000, cache size = 200).

$$\beta(r, v) = 1 - \pi_{N+1}(r, VC(v)). \tag{11}$$

The value of $\pi_{N+1}(r, VC(v))$ represents the miss probability of content c_r in the virtual cache of node v ($VC(v)$). We can see from Figure 3 that the solutions of our algorithm converge quickly in the different tested configurations. The convergence speed depends on the content popularity and the network settings, but in general, it takes about 8-10 iterations for all the contents to converge.

3.5. Single cache model for LCD

The authors in [6] proposed a new caching scheme called Leave Copy down (LCD). Under the Leave Copy Down (LCD) scheme, a new copy of the requested object is cached only on the node that resides immediately below the location of the hit on the path to the requesting client. Compared to LCE, LCD moves the requested contents progressively from the origin server towards the clients, with each request advancing a new copy of the document one hop closer to the client. LCD aims to reduce the redundancy of the same items at multiple nodes by caching an object at one node at a time and to avoid the amplification of replacement errors. The conducted experiments in [6] had shown the efficiency and good performance of LCD under different configurations and in addition, it is an easy scheme to implement that does not need additional overhead. If we model the LCD caching strategy using MACS, the value of $\beta(r)$ in this case will be the hit probability of content c_r in the next-hop cache. Using equations (10), the $\beta(r)$ of a node v is equal to:

$$\beta(r, v) = 1 - \pi_{N+1}(r, NH(v)). \tag{12}$$

The value of $\pi_{N+1}(r, NH(v))$ depicts the miss probability of c_r in the next-hop or parent cache of v (i.e. $NH(v)$). We can see from Figure 4 that the solutions of our algorithm converge quickly in the different tested configurations (it takes about 8-10 iterations for all the contents to converge).

4. Multiple caches system

Following common practice [12] [14], we assume in this work that after a cache miss and when a content is decided to be cached by a node, it will be downloaded instantaneously. Let's consider a system of multiple CCN nodes where the contents are forwarded according to the Shortest Path Routing (SPR) algorithm [27]. With SPR, when a client's interest cannot be satisfied by a node, it is forwarded along the shortest path to the closest permanent copy of the requested content. In this case, each node has to take into account, in addition to the local requests, the interests that come from other nodes due to a cache miss (we denote this stream of interests by "miss stream" or *MS*). The outgoing miss stream ratio from a node u of a content c_r is equal to

$$MS(r, u) = req(r, u) \quad \pi_{N+1}(r, u), \tag{13}$$

where $req(r, u)$ is the total proportion of requests for c_r received by u and $\pi_{N+1}(r, u)$ is the miss probability of content c_r at u. In CCNs, the interests for the same object received by a node will be aggregated and only the first one is sent to the next nodes. This feature should be considered when computing the total miss stream received by a node having more than one child node. The incoming miss stream ratio for an object with a rank r at a node v that we denote by $\eta_r(v)$, is equal to

$$\eta_r(v) = \sum_{u:NH(u)=v} \Big(MS(r, u) \prod_{w \neq u:NH(w)=v} (1 - MS(r, w)) \Big). \tag{14}$$

The set $\{u : NH(u) = v\}$ represents the nodes having v as the next hop in the shortest path toward the source. The value $1 - MS(r, w)$ represents the case where an interest sent from the node w is discarded because it was already received by another node. If we consider a multi-cache network where the requests aggregation feature is not present, $\eta_r(v)$ in this case will be simply equal to

$$\eta_r(v) = \sum_{u:NH(u)=v} MS(r, u). \tag{15}$$

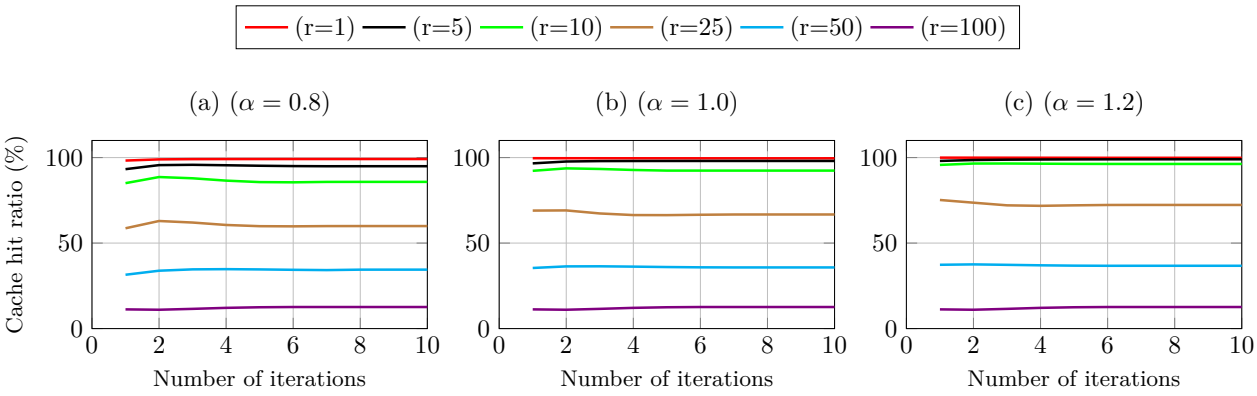

Figure 4: Cache hit ratio vs number of iterations in the LCD fixed-point iteration of contents with various popularities under a single node (catalog = 20000, cache size = 200).

Now, when dealing with an interconnected network of caching nodes, the probability that a node v will receive a request for c_r will no longer be p_r, but another value that we denote as p'_r, which will take into account in addition to the local requests, the interests due to a cache miss from previous nodes. For each node v, this value is equal to

$$p'_r = \frac{p_r + \eta_r(v)}{\sum_{k=1}^{R}(p_k + \eta_k(v))} = \frac{p_r + \eta_r(v)}{1 + \sum_{k=1}^{R}\eta_k(v)}. \tag{16}$$

In other words, p'_r represents here the proportion of requests received for c_r coming either from clients directly attached to the node (i.e. p_r) or from previous nodes (i.e. $\eta_r(v)$) over the total requests received by the node v for all the items. In the case when a CCN node doesn't have local requests, p'_r will be then equal to

$$p'_r = \frac{\eta_r(v)}{\sum_{k=1}^{R}\eta_k(v)}. \tag{17}$$

Consider again the previous Markov chain (see Figure 2). For every node v, we can compute the stationary state probabilities as we did in the case of a single node by replacing p_r with p'_r:

$$\begin{cases} \pi_1(r) = \dfrac{p'_r(1 + (\beta(r) - 1)\pi'_{N+1}(r))}{1 - (\gamma'_1(r) - \gamma_1(r))}, \\ \pi_2(r) = \gamma'_2(r)\pi_2(r) + (1 - \gamma'_1(r))\pi_1(r), \\ \pi_i(r) = \dfrac{(1 - p'_r - \gamma'_{i-1}(r))\pi_{i-1}(r)}{1 - \gamma'_i(r)}, 3 \leq i \leq N+1, \\ \sum_{i=1}^{N+1}\pi_i(r) = 1. \end{cases} \tag{18}$$

As we mentioned in previous section, the cache hit probability of a content with popularity r is equal to $1 - \pi_{N+1}(r)$. To compute the cache hit performance of a multi-cache system operating under LCE or 2Q, we start by treating the leaf nodes of the network since in our model each node needs to know all the incoming stream of requests, including those received due to a cache miss from a previous node. Starting from the leaves, we go through the core nodes of the network until arriving at the source (or root) node where the permanent copies of the catalog's objects are attached.

When a multi-cache system is operating under the LCD strategy, we cannot produce the steady-state hit rates in a bottom up way, like we did with the previous schemes. This is due to the bidirectional dependency a cache's state has with its upstream or downstream node (and vice versa). When modeling LCD and to compute the cache hit, each node in the network needs to have the incoming stream of requests received from previous nodes due to cache misses. At the same time, it is necessary for each node to know the cache hit rate of the upstream node in order to decide whether the object should be cached or not. To resolve this dependencies, we use a fixed-point iteration method. As a start, we compute the different incoming streams of all nodes assuming that the network operates under the LCE strategy. Since the root node represents the origin server containing all the available contents in network, its cache hit probability is then equal to one for all the items. Therefore, we can in a top down manner compute the hit ratio of the different caches starting from the root node and going down to the rest of the network using the request streams obtained with LCE. Then, we can repeat these steps using, each time, the new obtained values of equilibrium hit probabilities to recalculate the request streams and then, deduce the different hit ratios until their convergence.

5. Model evaluation

5.1. Simulation environment

In order to evaluate the accuracy of our proposals, we compared the analytical models presented in the previous section with the results of simulations under *ccnSim* [28], which is a discrete-event and a chunk-level simulator

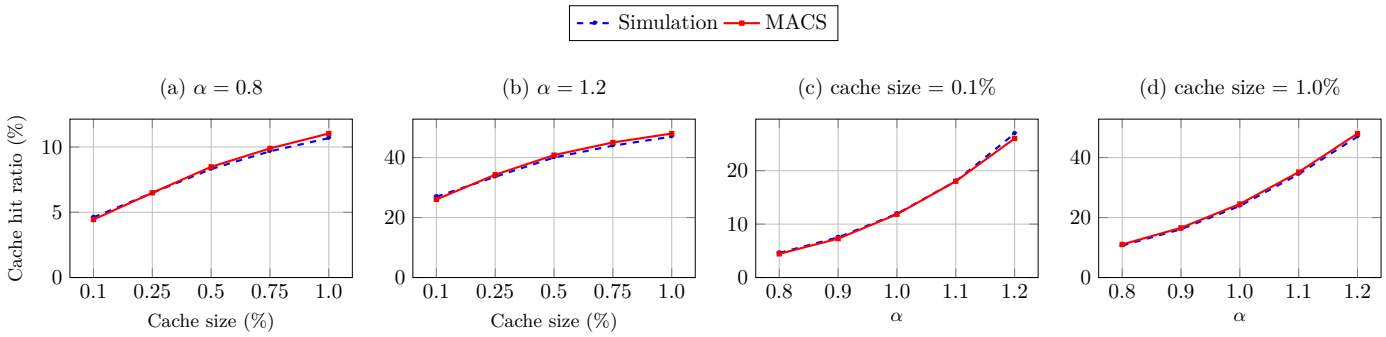

Figure 5: Total hit rate of the network using the 2Q scheme.

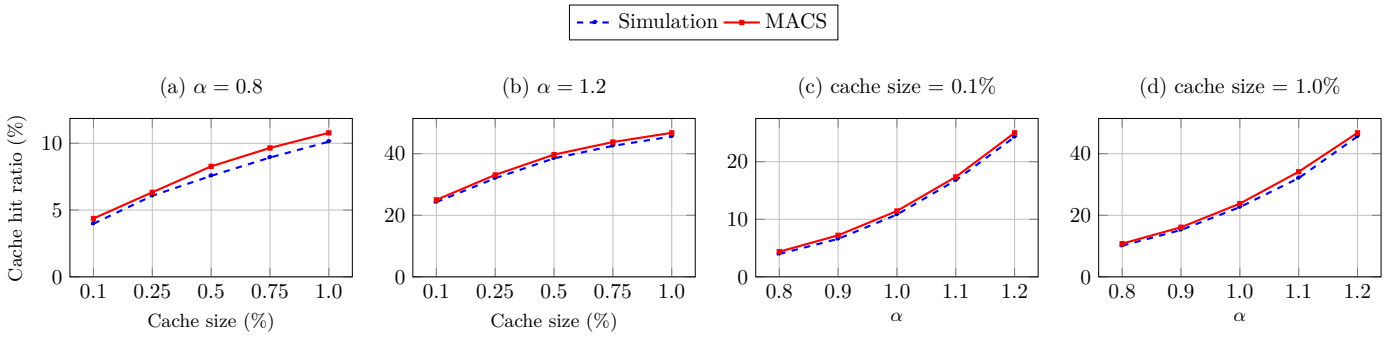

Figure 6: Total hit rate of the network using the LCD scheme.

for CCN networks. The accuracy of MACS, compared to the simulation results, can be affected by many parameters. Our focus on the conducted experiments was on the following key settings: cache size and Zipf law's skew distribution value. As for the network topology, a complete binary tree of 31 nodes was chosen to validate our model [1].

We measured during the tests three metrics: cache hit rate, content provider load and distance reduction ratio. The cache hit rate metric represents the ratio of requests that were served by the caches over the total number of requests sent in the network. The content provider load is defined as the proportion of requests not served by the intermediate caches of the network and thus, retrieved from the main source of contents. The distance reduction ratio represents the average gain obtained in terms of distance that an interest travels before finding a copy of the requested object. The content provider load and the distance reduction ratio (denoted respectively as CPL and DRR) are both expressed as a percentage and are computed as follows:

$$CPL = \Big(\prod_{i=1}^{s} P_{miss}(i)\Big) \times 100,$$

$$DRR = \frac{Dist(s) - \sum_{i=1}^{s} \Big(\big(\prod_{j=1}^{i-1} P_{miss}(j)\big) \times P_{hit}(i) \times Dist(i)\Big)}{Dist(s)} \times 100.$$

[1] Realistic network topologies can be also used as we did in [4].

The values $P_{miss}(i)$ and $P_{hit}(i)$ used here represent respectively the cache hit and cache miss of a network's node v_i. The expression $Dist(i)$ depicts the distance from where the clients requests were generated to the node v_i. The index s represents the nearest node v_s to the clients where a permanent copy of the contents is available (i.e. content provider). In the definition of CPL and DRR and for sake of clarity, we supposed that the network is formed by a line of nodes numbered from 1 to s where the clients are attached to node v_1 and the content repository is located at node v_s. It should be noted that of course, we can compute these metrics in any type of network topology.

In the simulation settings, we considered a catalog of contents containing 20,000 1-chunk sized objects whose popularity distribution follows the Zipf's law. Permanent copies of the available contents are hosted on one repository attached to the root node of the network. We set a uniform cache store capacity on the CCN nodes, which was defined as a proportion of the catalog size. Different simulations were conducted with a cache store size varying from 0.1% to 1.0% of the catalog capacity. The clients, attached to the network's leaves, generate requests according to a Poisson process with a rate per client corresponding to one request per second (each client representing an aggregate of users). We tested also different values of the Zipf law's skew parameter α going from 0.8 to 1.2. As shown in many studies [26], these two values correspond to the Zipf popularity exponent in the case of User Generated Content (UGC) and Video on Demand (VoD), respectively.

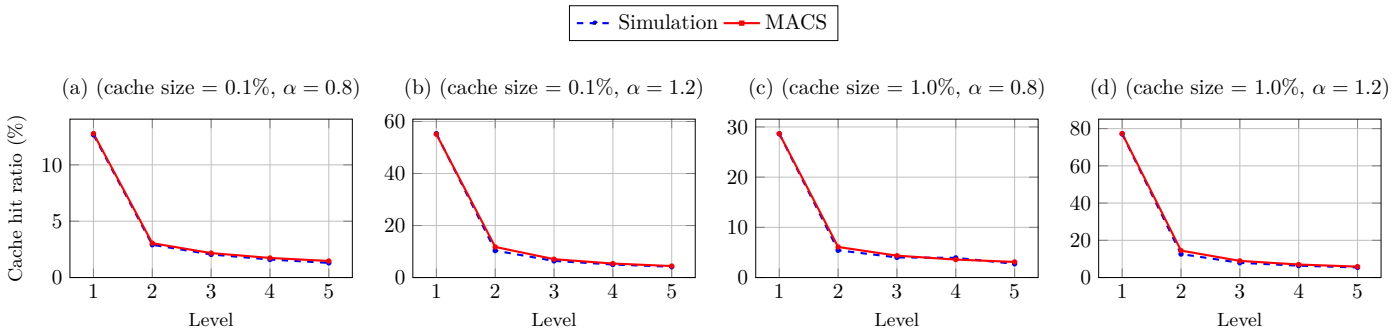

Figure 7: Cache hit ratio at different layers of the network using the 2Q scheme.

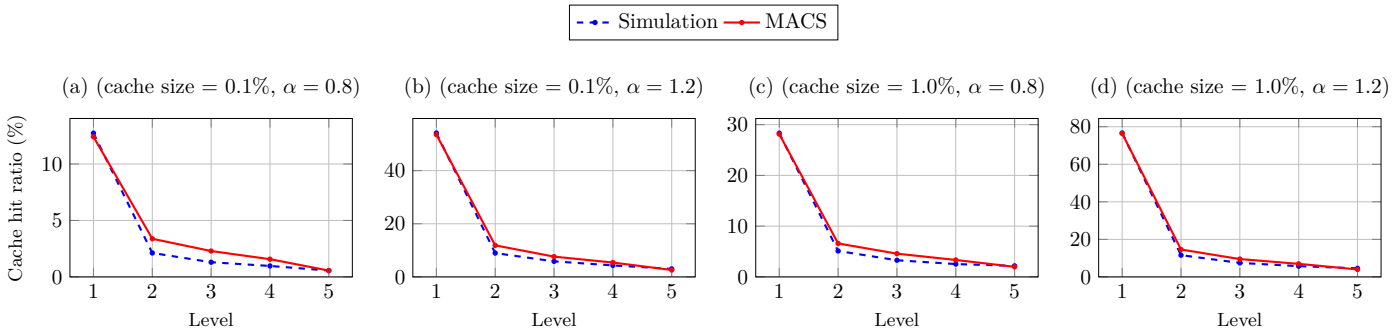

Figure 8: Cache hit ratio at different layers of the network using the LCD scheme.

The Least Recently Used (LRU) scheme is used as a cache replacement policy and the virtual buffer capacity used in the 2Q algorithm is equal to the main cache size, to see the model accuracy with different values of the virtual cache size. Next, we will expose and compare the cache hit results obtained with our analytical model and with the ccnSim simulation tool. The simulation results, shown in the graphs, depict the mean values taken over 30 runs, where 10^6 requests are sent in the network after it reaches its stability (i.e., all the caches become full). The error bars represent 99% confidence intervals.

5.2. Model results and analysis

In Figures 5-6, we plot the network's total hit rate as a function of the cache size and the Zipf's law skew parameter α of 2Q and LCD models along with ccnSim. Our aim here is to try a large range of values in the network configurations and see the model's performance in terms of accuracy. The results from the charts show a good accuracy in estimating the overall cache hit performance of the network with various settings. Compared to 2Q, the error rate in LCD model is a little higher. Let's recall that the complexity of modeling LCD with MACS is higher, as it requires the addition of a fixed-point iteration method to compute the cache hit performance of a multi-cache system, as explained in the previous section, which may increase the error rate.

Figures 7-8 display the cache hit accuracy at different levels of the network topology using 2Q and LCD. Level one represents the leaf caches and level five being the root node. The models accuracy is slightly reduced at higher levels within the network, especially with the LCD model. The main cause of this inaccuracy is related to the estimation of the request streams due to cache miss. Indeed, at the leaf nodes, the requests received contain only those generated by the clients, which can be easily estimated. However, the incoming miss streams of the nodes located at the core of the network are more difficult to estimate since they are computed using the cache hit rate of previous nodes.

Figures 9-10 compare 2Q and LCD models, respectively, with the simulations, conducted under different scenarios within ccnSim, in terms of average cache hit ratio as a function of content popularity. For the sake of clarity, we considered in the graphs only the objects whose popularity goes from 1 to 500. We can see from the charts that our analytical models give in average an accurate hit rate for the whole range of item population with different network settings. When the cache size is set to a low value (0.1% of the catalog), the models performs better even with distinct values of α and for different types of content popularity (see Figures 9(a), 9(b), 10(a) and 10(b)). A slight accuracy reduction in the cache hit ratio per content is observed when the cache capacity is set to 1% of the catalog size (see Figures 9(c), 9(d), 10(c) and 10(d)). An increase of the cache size means more states to be considered in MACS for each content, which makes it harder to estimate the hit ratio of each item.

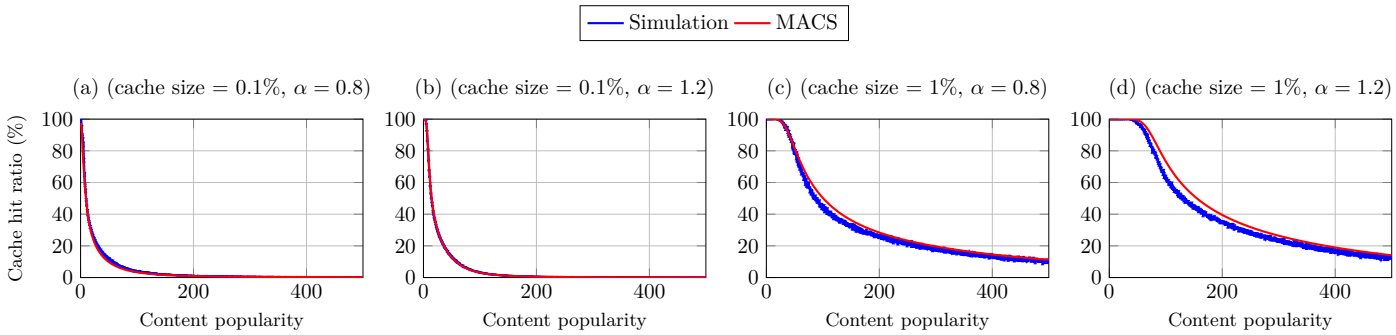

Figure 9: Total hit probability vs content popularity with different parameters and using the 2Q scheme.

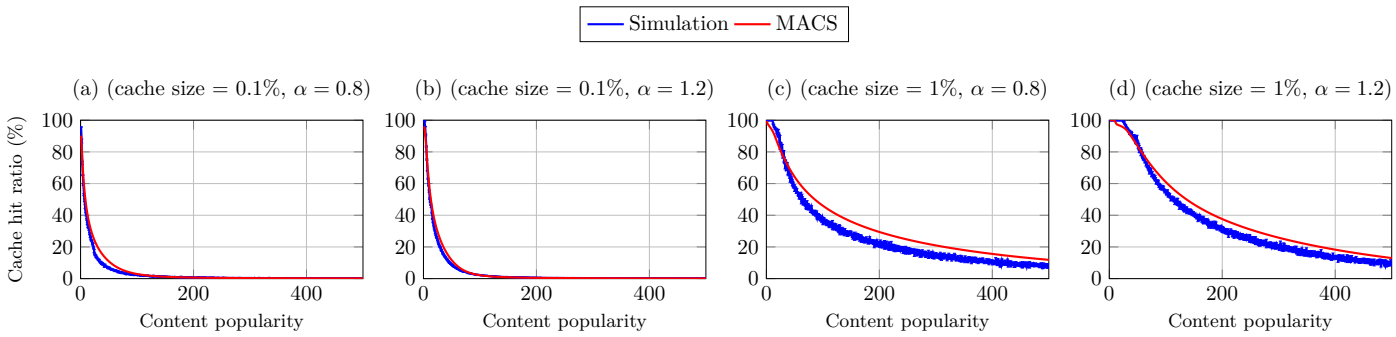

Figure 10: Total hit probability vs content popularity with different parameters and using the LCD scheme.

5.3. Caching algorithms comparison

We compare, in the following, the performance of the different caching strategies that we have modeled and analyzed in this paper or previous works: LCE, LCD, 2Q and Opt (only analytical results are shown). Opt represents the maximum cache hit ratio that we can achieve when the objective is to reduce the number of hops to retrieve contents. More specifically, if we consider a network where the caches have the same size N (in terms of number of items that can be stored) and the contents popularities are known in advance, then Opt consists on caching in the nodes located at 1-hop from the clients the most N popular contents of the catalog $(c_1, ..., c_N)$. Then, the second N most popular items $(c_{N+1}, ..., c_{2N})$ will be cached at the 2-hop nodes, etc. The results of Opt shown in the graphs were obtained using MACS by fixing to one the value of $\beta(r)$ for the contents that should be cached at each node and $\beta(r)$ is set to zero for the other items. The cache hit metric is an indicator of caching efficiency in multi-cache networks and generally, as the cache hit of each node increases, the network performance becomes better. Particularly, in traditional network topologies, we have many access caches and fewer and fewer core caches (a tree topology is a good example of this type of network). In this type of topology, access caches are linked to clients and often have similar content in the same region and the total cache hit is calculated as the average hit of the different caches in the network. As there is more caches at the access layer level, improving the overall cache hit would first require improving the hit of the access caches. This approach seems to be appropriate since it consists on bringing content closer to users and thus improving their quality of experience. This also has the advantage of reducing traffic in the operator's network. This strategy represents the main objective behind LCD and 2Q. For this reason, and when we did the comparison in terms of cache hit ratio, the target of Opt was to maximize the hop saving in order to cope with the aim of the caching schemes used in the state-of-the-art.

In Figure 11, we display the total hit rate of the network as a function of the α value (Zipf law) and the cache size using different caching schemes. The virtual cache size in 2Q was set to 20, which represents approximately the best tuning in the different use cases that were evaluated. We can see from the charts that LCD outperforms LCE, as it reduces the content redundancy in the network nodes and thus, more distinct objects are available in the caches. Each time a content is requested in LCD, it gets one-hop closer to the leaf nodes, which is an efficient way to detect and keep the popular items as closest as possible to the clients and to let the unpopular ones on higher levels of the network. Despite the efficiency of LCD, the results show the superiority of 2Q compared to the other caching schemes. Thanks to an effective filtering effect by the means of a virtual buffer, 2Q admits more popular contents into the cache than the other techniques.

As the cache size increases (Figures 11(a) and 11(b)), the performance difference between the evaluated caching

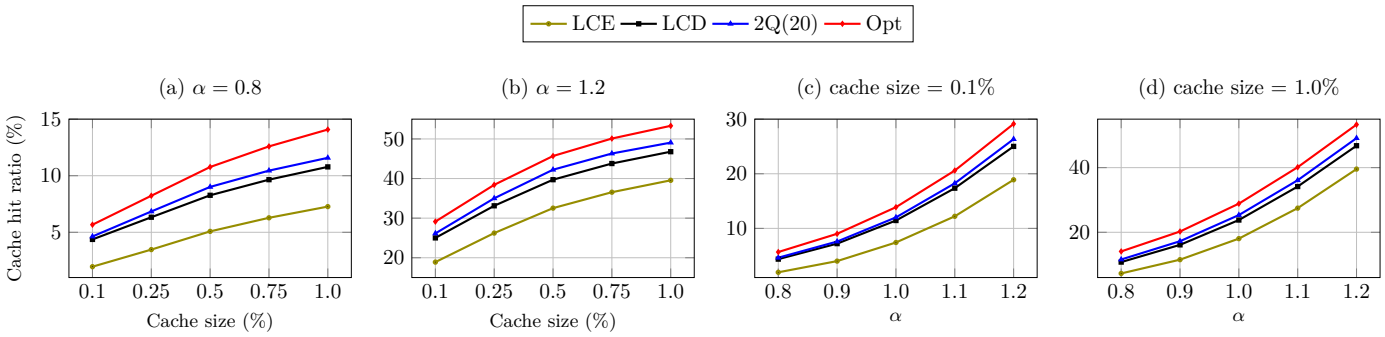

Figure 11: Total hit rate of the network with different parameters and various caching strategies.

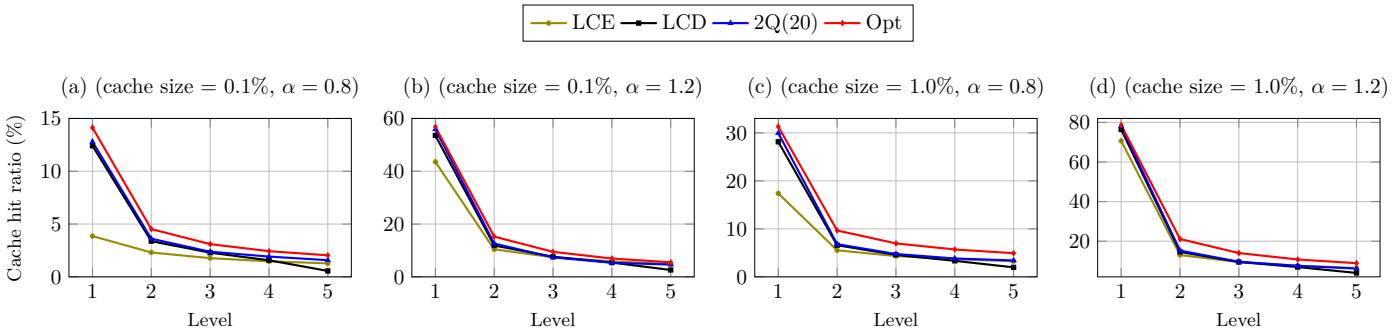

Figure 12: Total hit rate at different layers of the network with different parameters and various caching strategies.

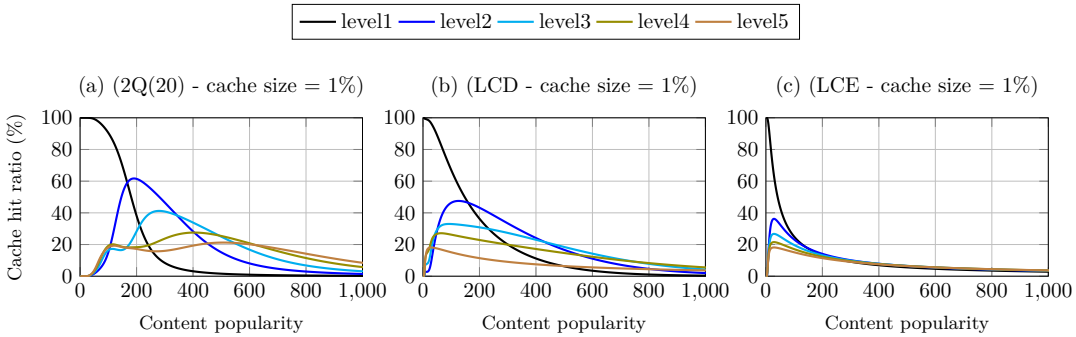

Figure 13: Cache hit ratio at different layers of the network with different parameters and various caching strategies.

schemes is reduced and gets closer to the optimal values. Increasing the storage capacity will diminish the impact of caching non popular items and the cache will become less affected by the adopted caching scheme. However, a limited cache size will increase the probability of discarding valuable contents to the benefit of the non popular ones. In this case, the efficiency of the decision caching strategy in accepting only the most popular items is crucial. For example, when α is set to 1.0 and the cache size is equal to 0.1% of the catalog, LCE achieves 53% of the optimal performance. However, if we set the cache size value to 1.0% of the catalog and we keep the same value of α, LCE performance reaches 63% of the optimal one.

The results also show a clear impact of the contents popularity distribution on the caching efficiency (Figures 11(c) and 11(d)). A high value of α (above one) means that fewer contents will receive most of the requests and, thus, decreasing the probability of caching many distinct unpopular items independently from the caching strategy. However, when α is set to a low value (below one), the contents popularity becomes more uniform, which makes it harder for the caching scheme to anticipate the most demanded items. For example, when the cache size is set 0.5% of the catalog and α is equal to 0.8, LCE achieves 47% of the optimal performance. In case where the α is set to 1.2 and without changing the cache size value, LCE performance reaches %71 of the optimal one.

Figure 12 displays the cache hit ratio at different layers of the networks with different settings and using different caching schemes. We can see from the different graphs

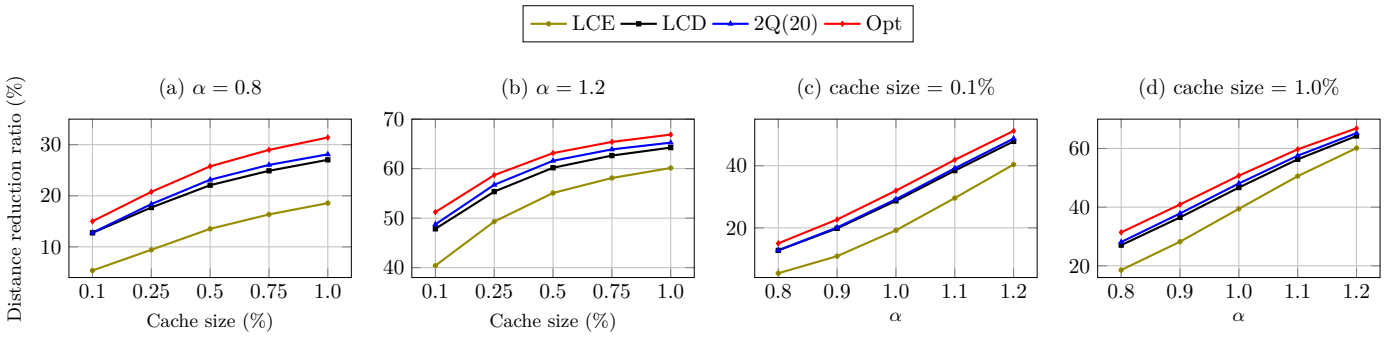

Figure 14: Average distance reduction ratio of the network with different parameters and various caching strategies.

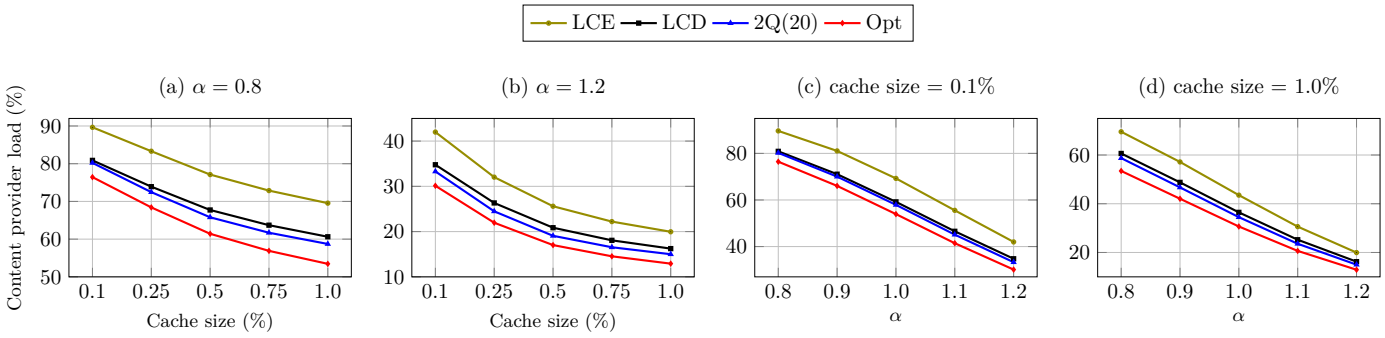

Figure 15: Content provider load of the network with different parameters and various caching strategies.

that the cache hit performance is much higher at the first level than the other ones, independently from the used caching scheme. The nodes at level one will be the first to receive the clients' requests, which increases their probability to serve the most popular contents demanded by the users. This will result in nodes at higher levels to cache less popular contents and thus, decreasing their performance. Compared to the total cache hit ratio of the network, it is clear that the first level nodes achieve most of it. The storage capacities located at higher levels of the network cannot be used efficiently and can be considered wasted resources. As we have highlighted previously during the analysis of the results depicted in Figure 11, the cache size and the Zipf law parameter α values have a significant impact on the caching efficiency, as it can be seen from the performance of the nodes at the first level of the network (Figure 12). As the cache size and *alpha* increase, caching the most popular objects and keeping them in the cache becomes easier, which will result in low performance difference between the considered caching schemes.

Figure 13 gives an idea about the contents getting the most hits at each level of the network, for different parameters and caching schemes. For the sake of clarity, the objects having a very low cache hit rate are not shown on the graphs. These results clearly confirms the superiority of LCD and, especially, 2Q over the other ones. Indeed, the graphs show clearly that 2Q succeeds in keeping at the nodes of level one more popular items than the other techniques. In LCD or LCE, the top most requested objects are not served exclusively by the first level nodes, which will increase the average distance to get contents. We can also see from the graphs how the popularity of cached contents and their hit ratio decrease as we go up on higher levels of the network. This can be explained by the fact that caching contents that already exists on previous nodes makes them useless, which increases the replacement errors of items in the cache and, thus, decreases the nodes performance.

In Figures 14-15, we display the distance reduction ratio and the content provider's load using different caching schemes. The results of Opt presented in Figure 14 are obtained by setting as its objective the reduction of distance to retrieve contents, as we explained it previously. In Figure 15, Opt was configured to minimize the usage of the content provider, which is achieved by caching the most popular items as near as possible to it. In other words, Opt in this case consists on caching in the nodes located at 1-hop from the content provider the most N popular contents of the catalog $(c_1, ..., c_N)$. Then, the second N most popular items $(c_{N+1}, ..., c_{2N})$ will be cached at the 2-hop nodes, etc. The graphs in Figures 14-15 confirm what we have obtained in the previous results. The performance of 2Q and LCD are very close, but 2Q remains always slightly better than LCD. Besides, the results indicate that increasing the cache size improves of course the network performance, but not in a uniform manner. In fact, doubling the cache size does not necessarily double the caching scheme efficiency. Now, if we compare the re-

sults of LCD and, especially, 2Q to Opt (see Figures 11, 12, 14 and 15), we can see that there is no much room left for improvement. In addition, a caching scheme that can reach or get closer to the theoretical optimal outcomes, in terms of network performance (i.e., cache hit ratio, content provider load, etc.), will necessarily increase the network operations overhead. The efficiency of the 2Q algorithm and its performance/overhead trade-off makes it a good candidate to be used as a caching strategy in CCN networks.

6. Conclusion

The Content-Centric Networking (CCN) paradigm is one of the most promising architectures for the future Internet. The in-network caching feature provided by CCN has a direct impact on the system performance, and it is important to analyze and evaluate the caching behavior in order to gain insights for optimized CCN caching schemes. We propose in this paper MACS, a Markov chain-based Approximation of CCN Caching Systems to estimate the cache hit probability under the popular LRU replacement policy. The model can be applied to different caching systems, not only to CCN, and can be used to compute different performance metrics in addition to the cache hit, such as the content provider load and the distance reduction ratio. The versatility of MACS enables us to model and analyze different caching schemes like LCE, LCD and 2Q. The conducted experiments clearly show the accuracy of our model in estimating the cache hit rate of a multi-cache system and also indicate the efficiency of 2Q in terms of network performance, which makes it a good candidate to be used as a caching strategy. As regards to our ongoing work, we are investigating, in addition to the caching schemes, the impact of cache resources placement in CCN and in multi-cache systems in general.

Acknowledgment

We would like to thank Lannion-Trégor Community and Regional Council of Brittany for their financial support.

References

[1] Cisco, Cisco visual networking index: forecast and methodology, 2016–2021, White paper (Jun. 2017).

[2] V. Jacobson, D. Smetters, J. Thornton, M. Plass, N. Briggs, R. Braynard, Networking named content, in: Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies, 2009, pp. 1–12.

[3] G. Rossini, D. Rossi, A dive into the caching performance of content centric networking, in: Proceedings of IEEE CAMAD, 2012, pp. 105–109.

[4] H. Ben-Ammar, S. A. Chellouche, Y. H. Aoul, A Markov chain-based Approximation of CCN caching Systems, in: IEEE Symposium on Computers and Communications, 2017, pp. 327–332.

[5] H. Ben-Ammar, Y. Hadjadj-Aoul, G. Rubino, S. Ait-Chellouche, A versatile markov chain model for the performance analysis of CCN caching systems, in: IEEE Global Communications Conference (to appear), 2018.

[6] N. Laoutaris, H. Che, I. Stavrakakis, The LCD interconnection of LRU caches and its analysis, Performance Evaluation 63 (2006) 609–634.

[7] T. Johnson, D. Shasha, 2q: A low overhead high performance buffer management replacement algorithm, in: Proceedings of the 20th International Conference on Very Large Data Bases, 1994, pp. 439–450.

[8] F. Guillemin, T. Houdoin, S. Moteau, Volatility of youtube content in orange networks and consequences, in: IEEE International Conference on Communications, 2013, pp. 2381–2385.

[9] J. Kurose, Information-centric networking: The evolution from circuits to packets to content, Computer Networks 66 (2014) 112 – 120.

[10] M. Zhang, H. Luo, H. Zhang, A survey of caching mechanisms in information-centric networking, IEEE Communications Surveys Tutorials 17 (2015) 1473–1499.

[11] W. F. King, Analysis of paging algorithms, in: IFIP Congress, 1971, pp. 485–490.

[12] A. Dan, D. Towsley, An approximate analysis of the LRU and FIFO buffer replacement schemes, SIGMETRICS Performance Evaluation Review 18 (1990) 143–152.

[13] E. J. Rosensweig, J. Kurose, D. Towsley, Approximate models for general cache networks, in: IEEE INFOCOM Proceedings, 2010, pp. 1–9.

[14] H. Che, Y. Tung, Z. Wang, Hierarchical web caching systems: modeling, design and experimental results, IEEE Journal on Selected Areas in Communications 20 (2002) 1305–1314.

[15] C. Fricker, P. Robert, J. Roberts, A versatile and accurate approximation for LRU cache performance, in: Proceedings of the 24th International Teletraffic Congress, 2012, pp. 1–8.

[16] N. C. Fofack, P. Nain, G. Neglia, D. Towsley, Performance evaluation of hierarchical ttl-based cache networks, Computer Networks 65 (2014) 212 – 231.

[17] I. Psaras, R. Clegg, R. Landa, W. Chai, G. Pavlou, Modelling and evaluation of ccn-caching trees, in: Proceedings of the 10th International IFIP Conference on Networking, 2011, pp. 78–91.

[18] S. Tarnoi, K. Suksomboon, W. Kumwilaisak, Y. Ji, Performance of probabilistic caching and cache replacement policies for content-centric networks, in: 39th Annual IEEE Conference on Local Computer Networks, 2014, pp. 99–106.

[19] S. Tarnoi, V. Suppakitpaisarn, W. Kumwilaisak, Y. Ji, Performance analysis of probabilistic caching scheme using markov chains, in: 2015 IEEE 40th Conference on Local Computer Networks, 2015, pp. 46–54.

[20] M. Garetto, E. Leonardi, V. Martina, A unified approach to the performance analysis of caching systems, ACM Trans. Model. Perform. Eval. Comput. Syst. 1 (2016) 1–20.

[21] N. Gast, B. V. Houdt, Ttl approximations of the cache replacement algorithms lru(m) and h-lru, Performance Evaluation 117 (2017) 33 – 57.

[22] E. J. O'Neil, P. E. O'Neil, G. Weikum, The lru-k page replacement algorithm for database disk buffering, SIGMOD Rec. 22 (1993) 297–306.

[23] M. Chrobak, J. Noga, LRU is better than FIFO, Algorithmica 23 (1999) 180–185.

[24] E. Gelenbe, A unified approach to the evaluation of a class of replacement algorithms, IEEE Transactions on Computers C-22 (1973) 611–618.

[25] S. Podlipnig, L. Böszörmenyi, A survey of web cache replacement strategies, ACM Comput. Surv. 35 (2003) 374–398.

[26] C. Fricker, P. Robert, J. Roberts, N. Sbihi, Impact of traffic mix on caching performance in a content-centric network, in: Proceedings IEEE INFOCOM Workshops, 2012, pp. 310–315.

[27] L. Wang, A. Hoque, C. Yi, A. Alyyan, B. Zhang, Ospfn: An ospf based routing protocol for named data networking, NDN Technical Report (Jul. 2012).

[28] R. Chiocchetti, D. Rossi, G. Rossini, ccnSim: An highly scalable ccn simulator, in: IEEE International Conference on Communications, 2013, pp. 2309–2314.